🧠 React Event Listeners – Explained with `Subscribe` and `Comment` Components

---

✅ `Subscribe.jsx` – Button Click Event Handler

📌 Code Summary:

```
import React from 'react';
import { useState } from 'react';
import './Subscribe.css';

const Subscribe = () ⇒ {
  const [subscriber, Subscriber_count] = useState(0); // State to keep track of
subscriber count

  function subscriber_manager() {
    Subscriber_count(subscriber + 1); // Increase subscriber count by 1
    alert('NEW SUBSCRIBER ADDED'); // Show alert on every new subscriber
  }

  return (
    <div>
      <button className="Subscribe-btn" onClick={subscriber_manager}>
        Subscribe!
      </button>
      <p>Subcribers:{subscriber}</p>
    </div>
  );
};
```

```
export default Subscribe;
```

🎯 Concepts Covered:

Concept	Description
`useState`	Used to hold subscriber count.
`onClick` event listener	Attached to button to trigger `subscriber_manager()` when clicked.
`alert()` + state update	Custom logic executed inside click handler.

✅ `Comment.jsx` – Input Event + Form Submit Handler

📌 Code Summary:

```
import React from 'react';
import './comment.css';

const Comment = () => {
  function handleSubmit(e) {
    e.preventDefault(); // Stops default page reload
    console.log('NEW POST ADDED');
  }

  function handleCommentbox(e) {
    console.log(e.target.value); // Logs current input value on every change
  }

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" onChange={handleCommentbox} />
      <button>Post</button>
    </form>
  );
};
```

```
export default Comment;
```

🎯 Concepts Covered:

Concept	Description
`onSubmit` on `<form>`	Handles form submit.
`preventDefault()`	Prevents page reload (default form behavior).
`onChange` on `<input>`	Fires on every keystroke to get live input.
`e.target.value`	Holds current text inside input field.

✅ `App.jsx` – Using Both Components

📌 Code Summary:

```
import { useState } from 'react';
import './App.css';
import Subscribe from './components/Subscribe';
import Comment from './components/Comment';

function App() {
  const [count, setCount] = useState(0);

  return (
    <>
      <Subscribe />
      <Comment />
    </>
  );
}

export default App;
```

🎯 Concepts Covered:

Concept	Description
Component usage	Shows how to reuse both `Subscribe` and `Comment` inside a parent app.
Separation of logic	Each component has its own event logic, making code modular and maintainable.

🧾 Final Summary (For Notes Title):

📌 React Event Listeners – Button Click, Form Submit & Input Change

- React uses JSX-style event listeners like `onClick`, `onSubmit`, `onChange`.
- Functions for handling these events are kept inside the same component.
- Use `e.preventDefault()` inside `onSubmit` to prevent unwanted reload.
- Use `useState()` when you need to store and update dynamic data (like subscriber count or input field value).
- Clean and modular structure keeps each component's logic independent and easy to manage.
